Name:

```
Apurba Roy Ajay ID:2018-3-60-063


Shahida Sultana ID:2018-1-60-117


Naeem Mizan     ID:2018-3-60-053
```

```
import numpy as np
import cv2
import os
import random
import matplotlib.pyplot as plt
```

```
DIRECTORY= r"/content/drive/MyDrive/CSE475/project2/leaf"
CATAGORIES= ['Strawberry_fresh','Strawberry_scrotch']
```

Data Description:

We use a Strawberry leaf related data for predicting is it fresh leaf or scrotch leaf.there are two features:Strawberry fresh and Strawberry scrotch.

In this project we create some image classification model.these are-

1.CNN-The Convolutional Neural Network (CNN)is a subtype of Neural Networks that is mainly used for applications in image and speech recognition. Its built-in convolutional layer reduces the high dimensionality of images without losing its information.

2.VGG16-VGG-16 is a kind of convolutional neural network that is 16 layers deep.

Data Preprocessing

In a dataset, there may be has some unwanted values, null values and unfit values.So, at first we should fixed this otherwise the model doesn’t work properly.

Pre-processing for CNN Network

```
data=[]

for categories in CATAGORIES:
    folder=os.path.join(DIRECTORY,categories) # joining the dataset
    label=CATAGORIES.index(categories)


    for img in os.listdir(folder):
        img=os.path.join(folder,img)
        img_arr=cv2.imread(img)
        img_arr=cv2.resize(img_arr,(100,100)) #resize images

        data.append([img_arr,label])
```

```
data
```

We have to suffle our dataset. Otherwise, if we choose e values in a sequential way, the model cannot predict properly.

```
random.shuffle(data)
```

Define feature and lebel

```
x=[]
y=[]
```

```
for features,label in data:
    x.append(features)
    y.append(label)
```

To turn our image into an array, we use CV2.

```
X= np.array(x)
Y=np.array(y)
```

```
x
```

```
X=X/255 #for simplicity in the equation
```

```
X
```

```
             [[0.65098039, 0.59215686, 0.62352941],
              [0.63137255, 0.57254902, 0.60392157],
              [0.64705882, 0.58823529, 0.61960784],
              ...,
              [0.49411765, 0.43921569, 0.47058824],
              [0.4627451 , 0.41176471, 0.44313725],
              [0.3254902 , 0.2745098 , 0.30588235]]],


            [[[0.65490196, 0.62745098, 0.6745098 ],
              [0.67058824, 0.64705882, 0.69019608],
              [0.66666667, 0.64313725, 0.68627451],
              ...,
              [0.77254902, 0.74901961, 0.79215686],
              [0.76470588, 0.74117647, 0.78431373],
              [0.8       , 0.77254902, 0.81568627]],

             [[0.68627451, 0.6627451 , 0.70588235],
              [0.64705882, 0.62352941, 0.66666667],
              [0.63529412, 0.61176471, 0.65490196],
              ...,
              [0.74901961, 0.7254902 , 0.76862745],
              [0.81568627, 0.79215686, 0.83529412],
              [0.77254902, 0.74901961, 0.79215686]],

             [[0.69803922, 0.6745098 , 0.71764706],
              [0.63137255, 0.60784314, 0.65098039],
              [0.70196078, 0.67843137, 0.72156863],
              ...,
              [0.78039216, 0.75686275, 0.8       ],
              [0.80392157, 0.77647059, 0.82352941],
              [0.81176471, 0.78823529, 0.83137255]],

             ...,

             [[0.69019608, 0.65882353, 0.70980392],
              [0.72941176, 0.70196078, 0.75294118],
              [0.72941176, 0.70196078, 0.75294118],
              ...,
              [0.82352941, 0.79607843, 0.82352941],
              [0.84705882, 0.81960784, 0.84705882],
              [0.8       , 0.77254902, 0.8       ]],

             [[0.65490196, 0.62745098, 0.67843137],
              [0.74117647, 0.71372549, 0.76470588],
              [0.76862745, 0.74117647, 0.79215686],
              ...,
              [0.78431373, 0.75686275, 0.78431373],
              [0.83529412, 0.80784314, 0.83529412],
              [0.80784314, 0.78039216, 0.80784314]],

             [[0.78431373, 0.75686275, 0.80784314],
              [0.65490196, 0.62745098, 0.67843137],
              [0.76078431, 0.73333333, 0.78431373],
              ...,
              [0.76862745, 0.74117647, 0.76862745],
              [0.77647059, 0.74901961, 0.77647059],
              [0.79607843, 0.76862745, 0.79607843]]]])
```

```
X.shape
```

```
(100, 100, 100, 3)
```

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D,MaxPooling2D,Flatten,Dense,Activation
```

There are four layer in CNN.We define all of them in here with a activation function

```
model=Sequential()
model.add( Conv2D(64,(3,3),input_shape=X.shape[1:],activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add( Conv2D(32,(3,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add( Conv2D(32,(3,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))


model.add(Flatten())

model.add(Dense(2,activation='softmax'))
```

```
model.compile(loss='sparse_categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
```

```
model.fit(X,Y,epochs=10,validation_split=0.1)
```

```
Epoch 1/10
3/3 [==============================] - 8s 294ms/step - loss: 0.6889 - accuracy: 0.5000 - val_loss: 0.6771 - val_accuracy: 0.6000
Epoch 2/10
3/3 [==============================] - 0s 22ms/step - loss: 0.6576 - accuracy: 0.6444 - val_loss: 0.6654 - val_accuracy: 0.9000
Epoch 3/10
3/3 [==============================] - 0s 22ms/step - loss: 0.6090 - accuracy: 0.8333 - val_loss: 0.6323 - val_accuracy: 0.8000
Epoch 4/10
3/3 [==============================] - 0s 22ms/step - loss: 0.5451 - accuracy: 0.8444 - val_loss: 0.5435 - val_accuracy: 1.0000
Epoch 5/10
3/3 [==============================] - 0s 21ms/step - loss: 0.4453 - accuracy: 0.8778 - val_loss: 0.4542 - val_accuracy: 0.8000
Epoch 6/10
3/3 [==============================] - 0s 21ms/step - loss: 0.3589 - accuracy: 0.9222 - val_loss: 0.4924 - val_accuracy: 0.6000
Epoch 7/10
3/3 [==============================] - 0s 21ms/step - loss: 0.3307 - accuracy: 0.8556 - val_loss: 0.4010 - val_accuracy: 0.8000
Epoch 8/10
3/3 [==============================] - 0s 22ms/step - loss: 0.2837 - accuracy: 0.8889 - val_loss: 0.2868 - val_accuracy: 1.0000
Epoch 9/10
3/3 [==============================] - 0s 21ms/step - loss: 0.2942 - accuracy: 0.8778 - val_loss: 0.2865 - val_accuracy: 0.8000
Epoch 10/10
3/3 [==============================] - 0s 21ms/step - loss: 0.2523 - accuracy: 0.8667 - val_loss: 0.2390 - val_accuracy: 0.8000
<keras.callbacks.History at 0x7fbd800bb850>
```

Double-click (or enter) to edit

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 98, 98, 64)        1792

 max_pooling2d (MaxPooling2D  (None, 49, 49, 64)       0
 )

 conv2d_1 (Conv2D)           (None, 47, 47, 32)        18464

 max_pooling2d_1 (MaxPooling  (None, 23, 23, 32)       0
 2D)

 conv2d_2 (Conv2D)           (None, 21, 21, 32)        9248

 max_pooling2d_2 (MaxPooling  (None, 10, 10, 32)       0
  2D)

  flatten (Flatten)           (None, 3200)              0

  dense (Dense)               (None, 2)                 6402

=================================================================
Total params: 35,906
Trainable params: 35,906
Non-trainable params: 0
_________________________________________________________________
```

```
from keras.preprocessing import image
import numpy as np

img_pred=image.load_img(r"/content/drive/MyDrive/CSE475/project2/leaf/Strawberry_fresh/02caa98d-1c74-43b3-b3ee-e8492998f82a___RS_HL 2090.JPG"

img_pred=image.img_to_array(img_pred)
img_pred=np.expand_dims(img_pred, axis=0)


rslt= model.predict(img_pred)

print(rslt)
if rslt[0][0]>rslt[0][1]:
    prediction="Strawberry_fresh"


else:
    prediction="Strawberry_scrotch"
print(prediction)
```

! 0s completed at 3:29 AM